



# Creación de Aplicaciones Python

## Lección 14: PyWebIO

# ÍNDICE

|   |          |
|---|----------|
| <b>Lección 14. – PyWebIO .....</b>                        | <b>2</b> |
| Presentación y objetivos .....                            | 2        |
| 1. Ejemplo 1 con PyWebIO : Input básica.....              | 3        |
| 2. Ejemplo 2 con PyWebIO : 3 elementos típicos .....      | 6        |
| 3. Ejemplo 3 con PyWebIO : Textarea .....                 | 8        |
| 4. Ejemplo 4 con PyWebIO : Salida de Texto básica.....    | 10       |
| 5. Ejemplo 5 con PyWebIO : Ventana emergente .....        | 12       |
| 6. Ejemplo 6 con PyWebIO : Ventana emergente [2] .....    | 13       |
| 7. Ejemplo 7 con PyWebIO : Put_collapse .....             | 14       |
| 8. Ejemplo 8 con PyWebIO : Put_Buttons.....               | 16       |
| 9. Ejemplo 9 con PyWebIO : Put_row / Put_column .....     | 17       |
| 10. Ejemplo 10 con PyWebIO : Style( ).....                | 19       |
| 11. Otros puntos a tratar y cierre de la asignatura ..... | 20       |
| 12. Puntos clave.....                                     | 21       |

# Lección 14. – PyWebIO


## PRESENTACIÓN Y OBJETIVOS

Para aquellas personas que les guste Streamlit existe otro Framework también muy sencillo de implementar, pero algo más versátil. Su nombre es PyWebIO, y a fecha de Abril 2021 es de las cosas más recientes y de gran calidad que existen. (En este caso 30 Abril 2020, fue la fecha de la “release 0.2” según su propia página web).

Su web es:

<https://pywebio.readthedocs.io/en/latest/>

Para explicarlo indicaremos inicialmente el código de un ejemplo, y las figuras siguientes serán pantallazos de la propia ejecución paso a paso.

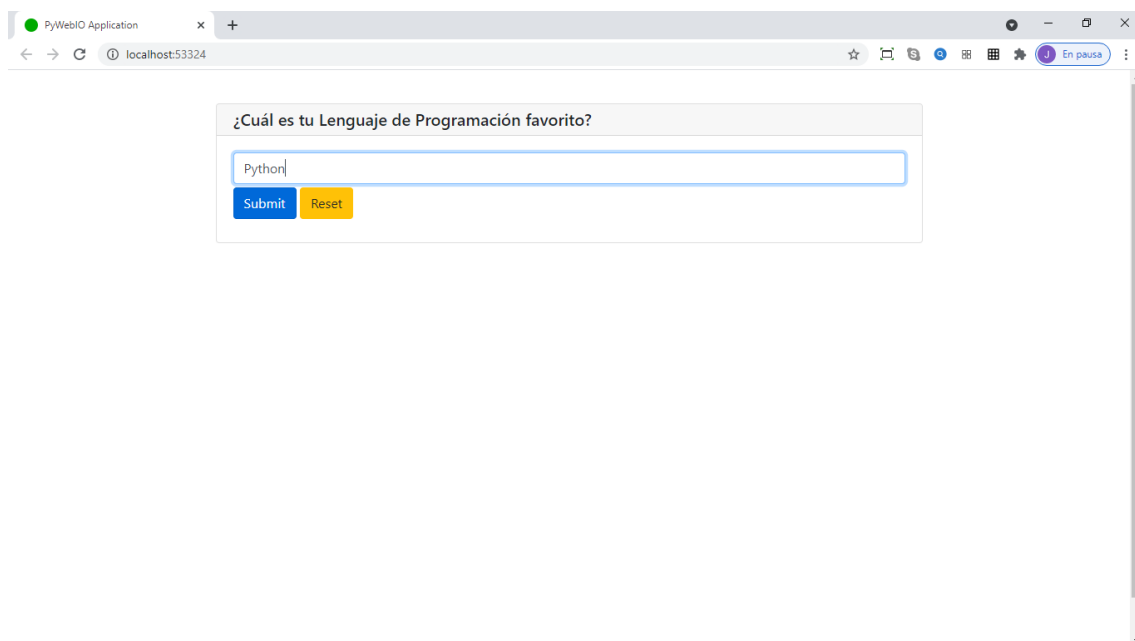
|   |   |
|---|---|
|  | <h3>Objetivos</h3> <ul style="list-style-type: none"><li>• Aprender los conceptos generales de PyWebIO, y de esta forma, ver la versatilidad que tiene.</li></ul> |
|---|---|

Comencemos !

## 1. EJEMPLO 1 CON PYWEBIO : INPUT BÁSICA

```
pywebio_1.py
1  • from pywebio.input import *
2  • from pywebio.output import *
3
4
5  # input 1 - texto
6  input("¿Cuál es tu Lenguaje de Programación favorito?")
7
8  # input 2 - números
9  input("¿Cuántos Frameworks de Python conoces?",
10 • | type=NUMBER)
11
12 # input 3 - Password
13 input("Escriba su contraseña",
14 • | type=PASSWORD)
```

Figura 1.1: Ejemplo básico con PyWebIO (parte 1)



PyWebIO Application x +

localhost:53324

¿Cuál es tu Lenguaje de Programación favorito?

Python

Submit Reset

Figura 1.2: Ejemplo básico con PyWebIO (parte 2)

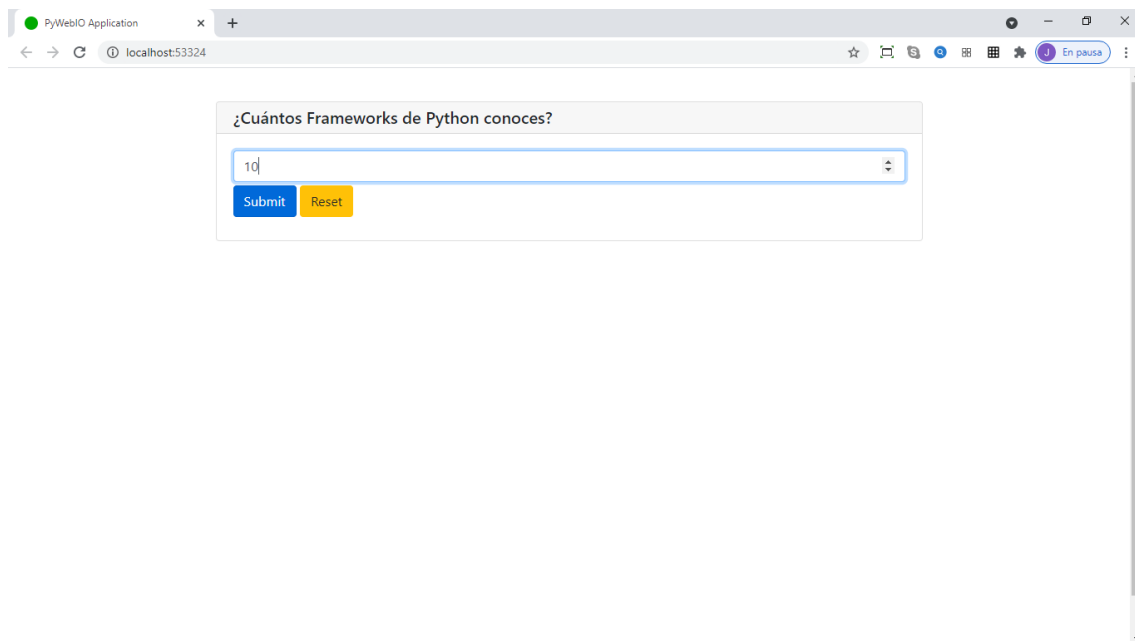


Figura 1.3: Ejemplo básico con PyWebIO (parte 3)

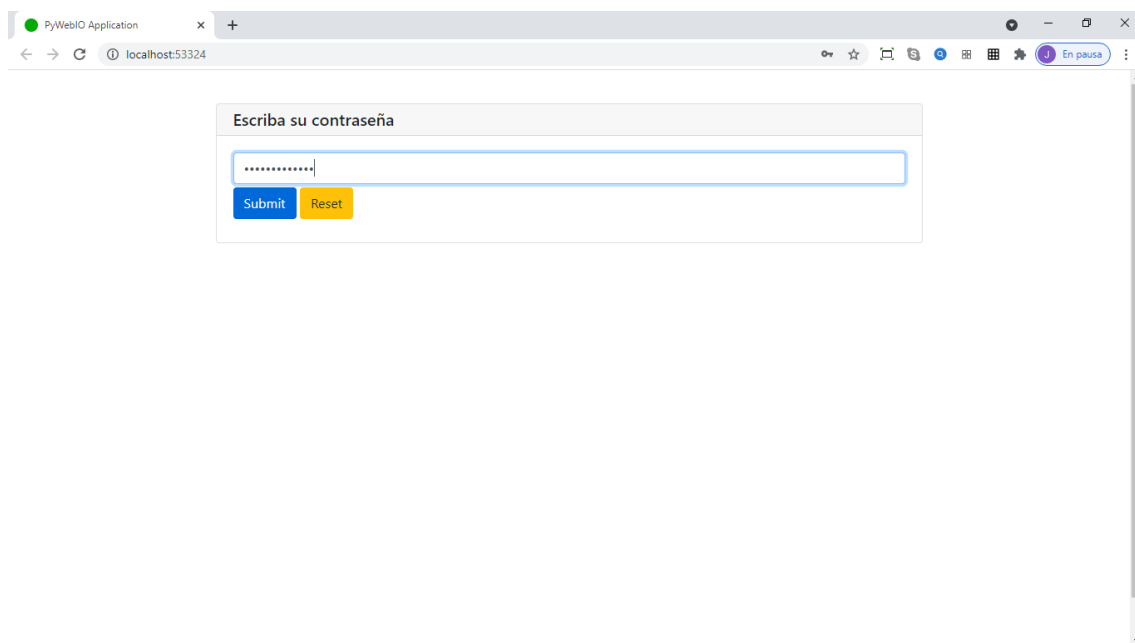
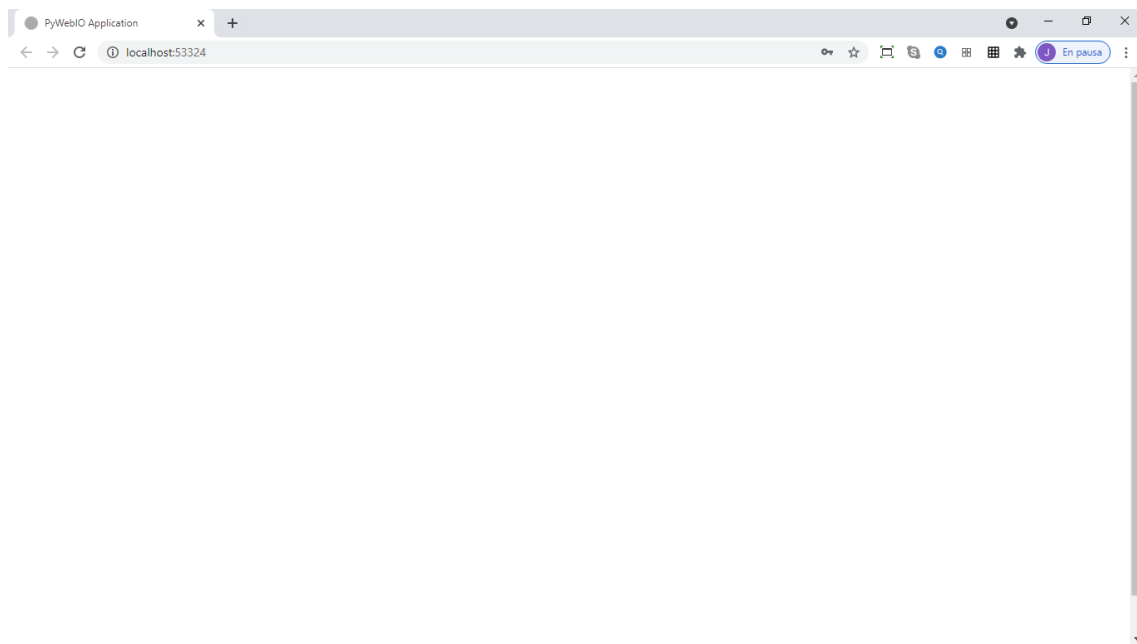


Figura 1.4: Ejemplo básico con PyWebIO (parte 4)



*Figura 1.5: Ejemplo básico con Streamlit (parte 5)*

De modo que tenemos una salida muy versátil que aparece según vamos añadiendo información y hacemos click en “submit”.

## 2. EJEMPLO 2 CON PYWEBIO : 3 ELEMENTOS TÍPICOS

```

pywebio_2.py
1  • from pywebio.input import *
2  • from pywebio.output import *
3
4
5  # radio
6  • radio("Selecciona un Framework GUI <radio button> :",
7         options=["PyQT", "Tkinter"])
8
9  # checkbox
10 • checkbox("¿Conoces el Framework Dash?",
11           options=["SI", "NO"])
12
13 # select
14 • select("Selecciona un Framework <select> :",
15         ["Django", "Streamlit", "PyWebIO"])

```

Figura 2.1: 3 elementos básicos en PyWebIO (parte 1)

PyWebIO Application x +

localhost:53452

Selecciona un Framework GUI <radio button> :

☐ PyQT

☒ Tkinter

Figura 2.2: 3 elementos básicos en PyWebIO (parte 2)

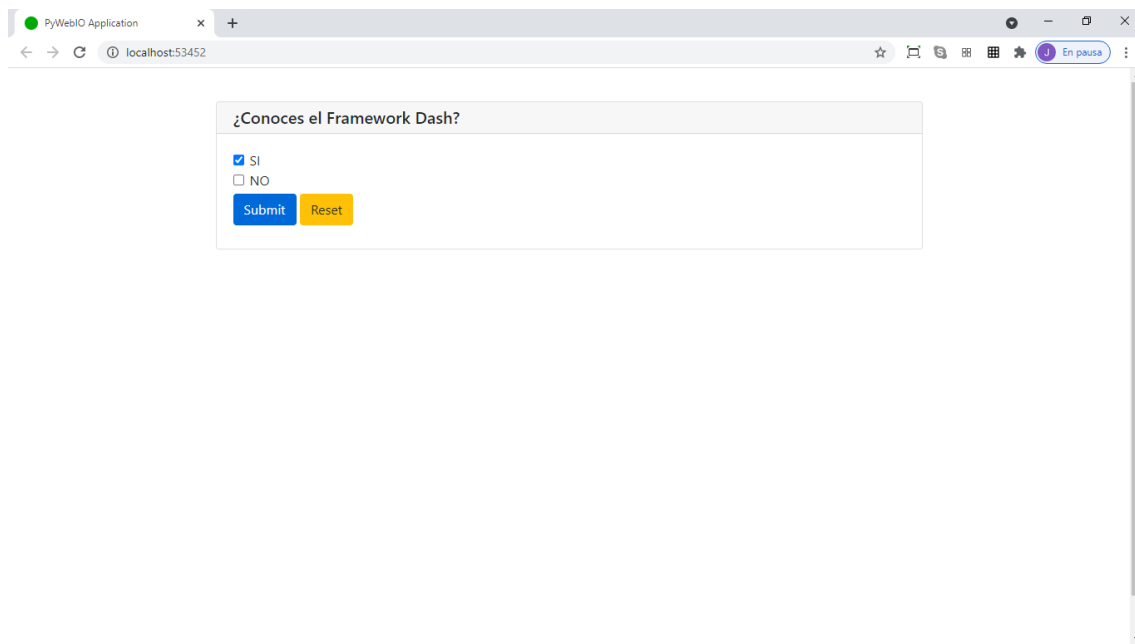


Figura 2.3: 3 elementos básicos en PyWebIO (parte 3)

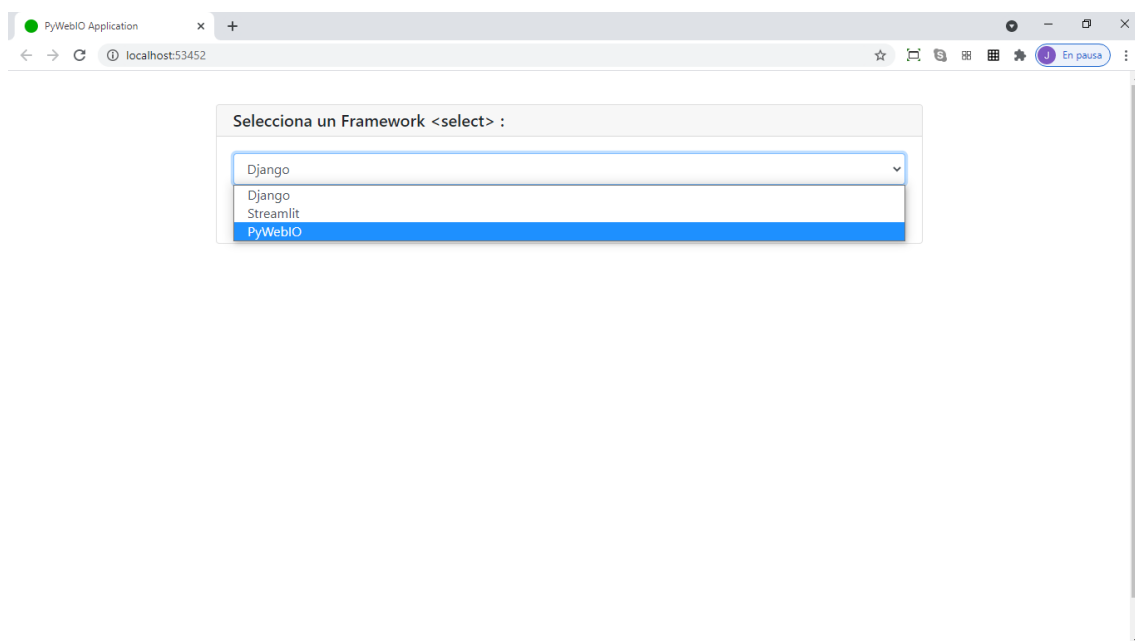


Figura 2.4: 3 elementos básicos en PyWebIO (parte 4)

Como veremos la ventana quedará en blanco al indicar “submit” en este caso.



### 3. EJEMPLO 3 CON PYWEBIO : TEXTAREA

```
pywebio_3.py
1  • from pywebio.input import *
2  • from pywebio.output import *
3
4
5  # textarea - 1
6  • textarea('Text Area',
7          rows=3,
8          placeholder='Escriba su texto aquí..')
9
10 # textarea - 2
11 • textarea('Code Edit',
12          code={
13              'mode': "python",
14              'theme': 'darcula',
15          },
16          value='import pandas as pd\n# Escriba aquí código en Python')
```

Figura 3.1: Textarea en PyWebIO (parte 1)

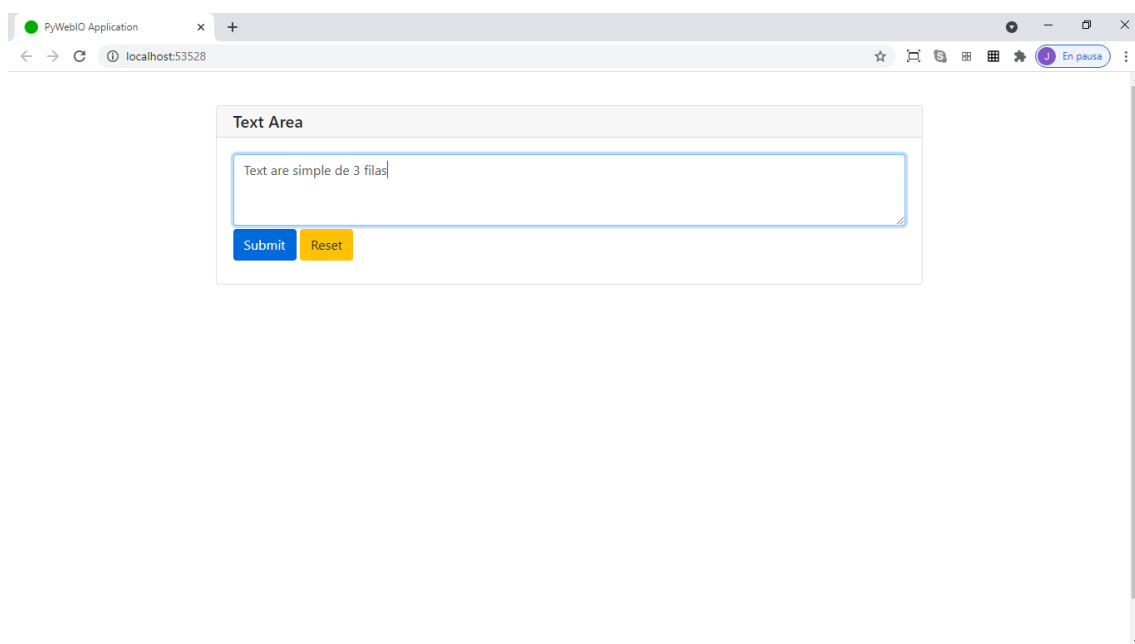


Figura 3.2: Textarea en PyWebIO (parte 2)

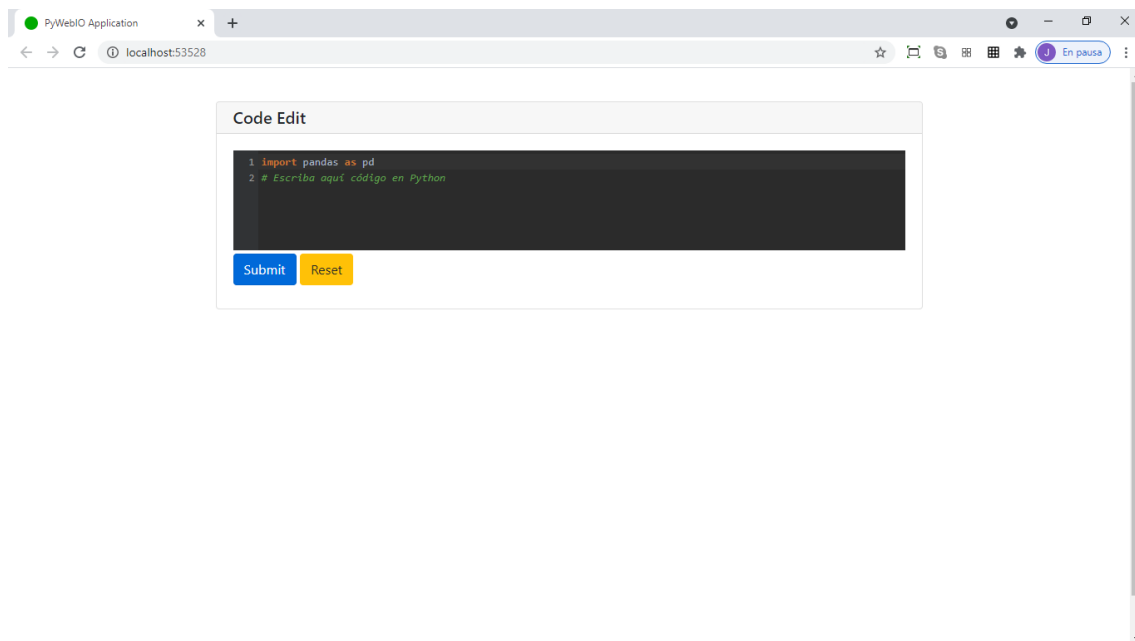


Figura 3.3: Textarea en PyWebIO (parte 3)

Que podríamos escribir código en Python, tal y como se puede apreciar.

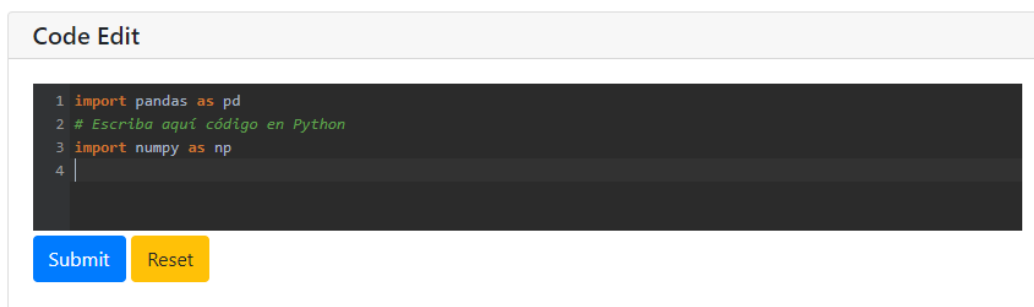


Figura 3.4: Textarea en PyWebIO (parte 4)

Al indicar “submit” nuevamente me dejará la aplicación en blanco.

## 4. EJEMPLO 4 CON PYWEBIO : SALIDA DE TEXTO BÁSICA

```
pywebio_4.py x
1  ● from pywebio.input import *
2  ● from pywebio.output import *
3
4
5  # Text Output
6  ● put_text("Con put_text realizamos salida de texto")
7
8  # Table Output
9  ● put_table([
10     ["Empresa", "Cotización (€)"],
11     ["X", "10"],
12     ["Y", "0.5"],
13     ["Z", "8.1"],
14 ])
15
16 # Markdown 1
17 ● put_markdown("~~Mensaje tachado~~")
18
19 # markdown 2
20 ● put_markdown("**Mensaje en negrita**")
```

Figura 4.1: Output básica en PyWebIO (parte 1)

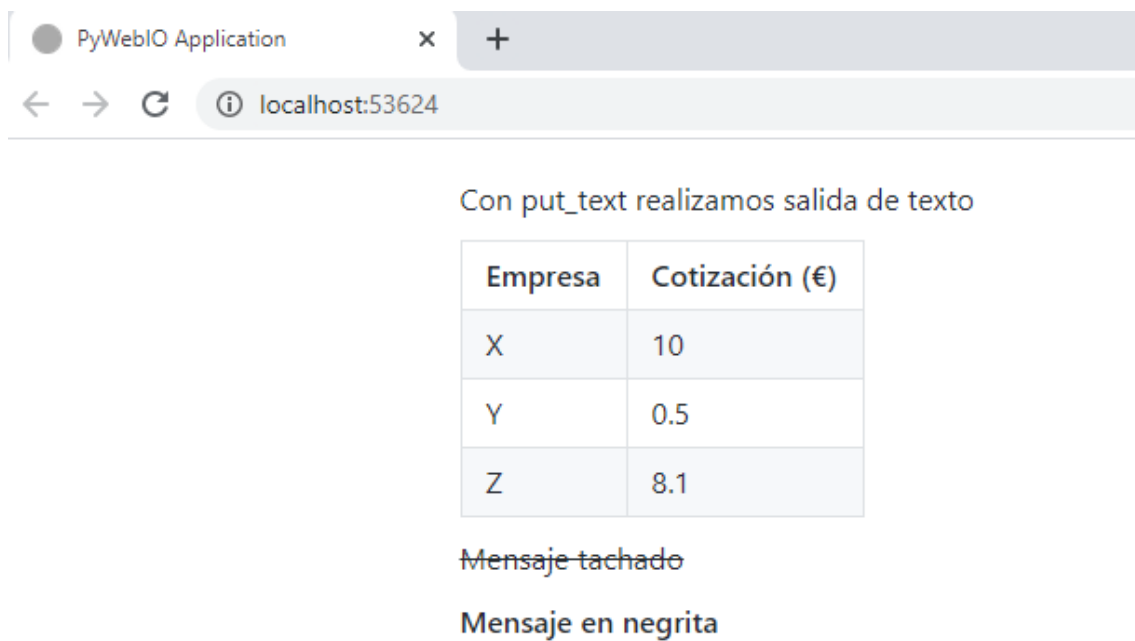


Figura 4.2: Output básica en PyWebIO (parte 2)

Que vemos cómo afecta la salida, toda impresa a un mismo tiempo.  
(Ya no son elementos “input”)

## 5. EJEMPLO 5 CON PYWEBIO : VENTANA EMERGENTE

```
pywebio_5.py
1  ● from pywebio.input import *
2  ● from pywebio.output import *
3
4
5  # ventana emergente (PopUp)
6  ● popup("Título de la ventana emergente",
7         "Contenido de la ventana emergente")
```

Figura 5.1: Popup en PyWebIO (parte 1)

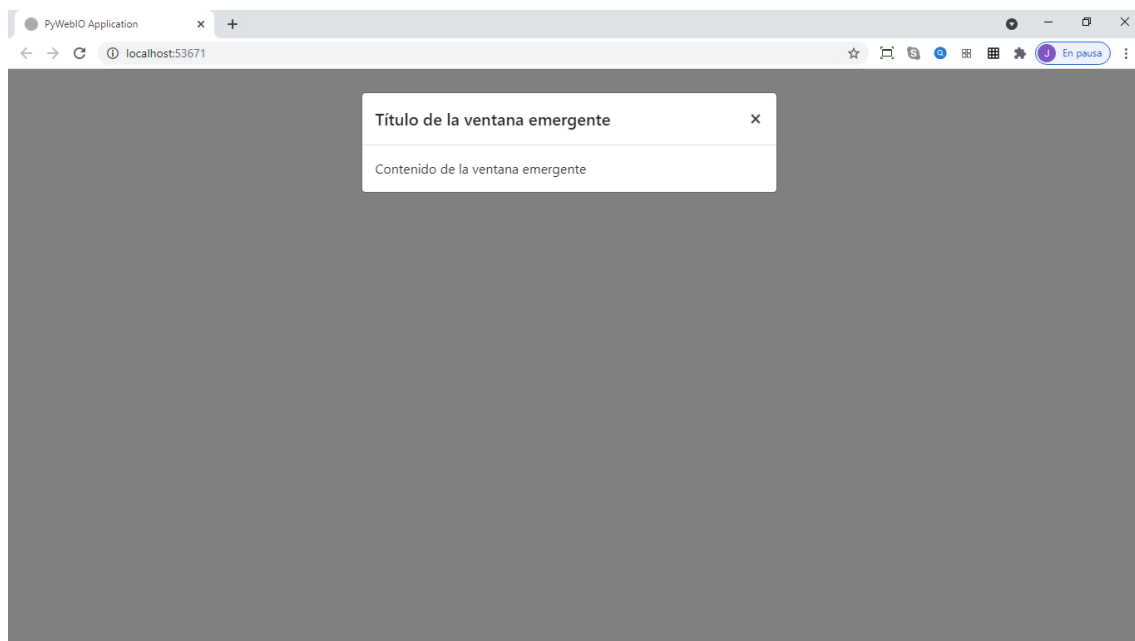


Figura 5.2: Popup en PyWebIO (parte 2)

## 6. EJEMPLO 6 CON PYWEBIO : VENTANA EMERGENTE [2]

```

pywebio_6.py
1  ● from pywebio.input import *
2  ● from pywebio.output import *
3
4
5  # ventana emergente 2
6  ● popup("Título de la ventana emergente",
7      [
8  ●      put_html("<h1>Contenido en H1</h1>"),
9  ●      put_html("<h2>Contenido en H2</h2>"),
10 ●      put_html("<h3>Contenido en H3</h3>"),
11 ●      put_html("<h4>Contenido en H4</h4>"),
12 ●      put_html("<h5>Contenido en H5</h5>"),
13 ●      put_html("<h6>Contenido en H6</h6>")
14 ]

```

Figura 6.1: Popup con HTML en PyWebIO (parte 1)

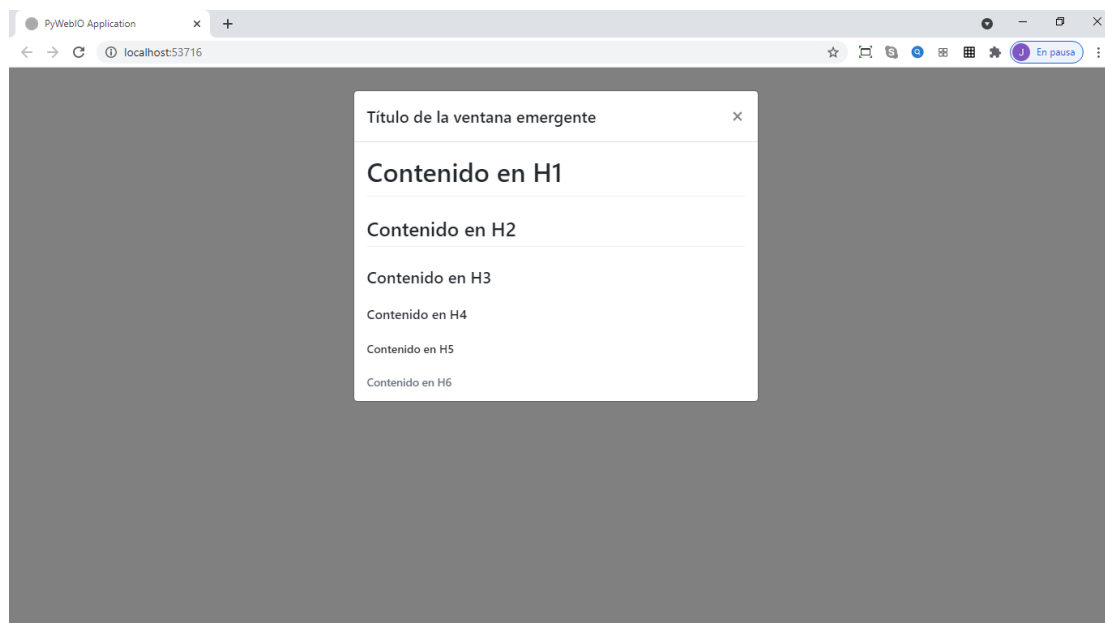


Figura 6.2: Popup con HTML en PyWebIO (parte 2)

## 7. EJEMPLO 7 CON PYWEBIO : PUT\_COLLAPSE

```
pywebio_7.py x
1  ● from pywebio.input import *
2  ● from pywebio.output import *
3
4
5  ● with put_collapse('Despliega el menú con 4 opciones:'):
6      for i in range(1, 5, 1):
7  ●          put_text("Opción: ", i)
```

Figura 7.1: Put\_collapse en PyWebIO (parte 1)

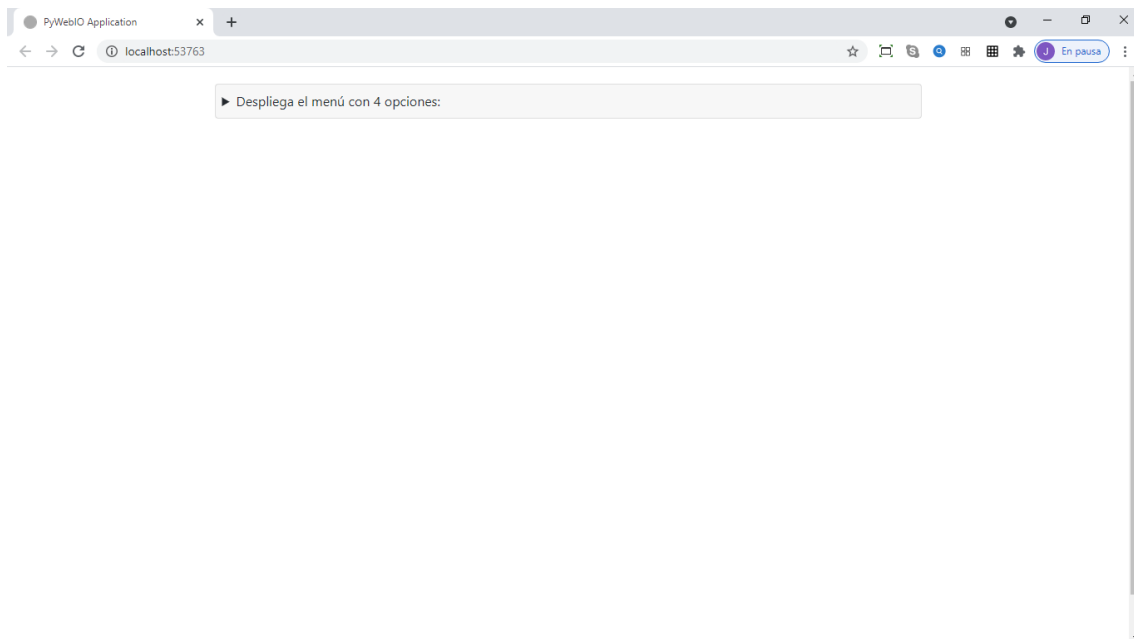
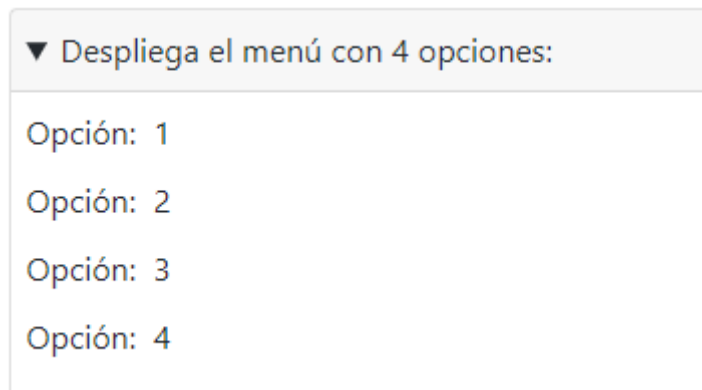


Figura 7.2: Put\_collapse en PyWebIO (parte 2)



*Figura 7.3: Put\_collapse en PyWebIO (parte 3)*



## 8. EJEMPLO 8 CON PYWEBIO : PUT\_BUTTONS

```

pywebio_8.py
1  from pywebio.input import *
2  from pywebio.output import *
3  import pywebio
4
5
6  def funcion_click(valor_boton):
7      if valor_boton == "Java" or valor_boton == "C++":
8          put_text(valor_boton, ": es un buen Lenguaje de Programación")
9      else:
10         # python
11         put_text(valor_boton, ": es absolutamente impresionante!")
12
13
14
15  put_buttons(['Python', 'C++', 'Java'], onclick=funcion_click)
16
17  pywebio.session.hold()

```

Figura 8.1: Put\_buttons en PyWebIO (parte 1)

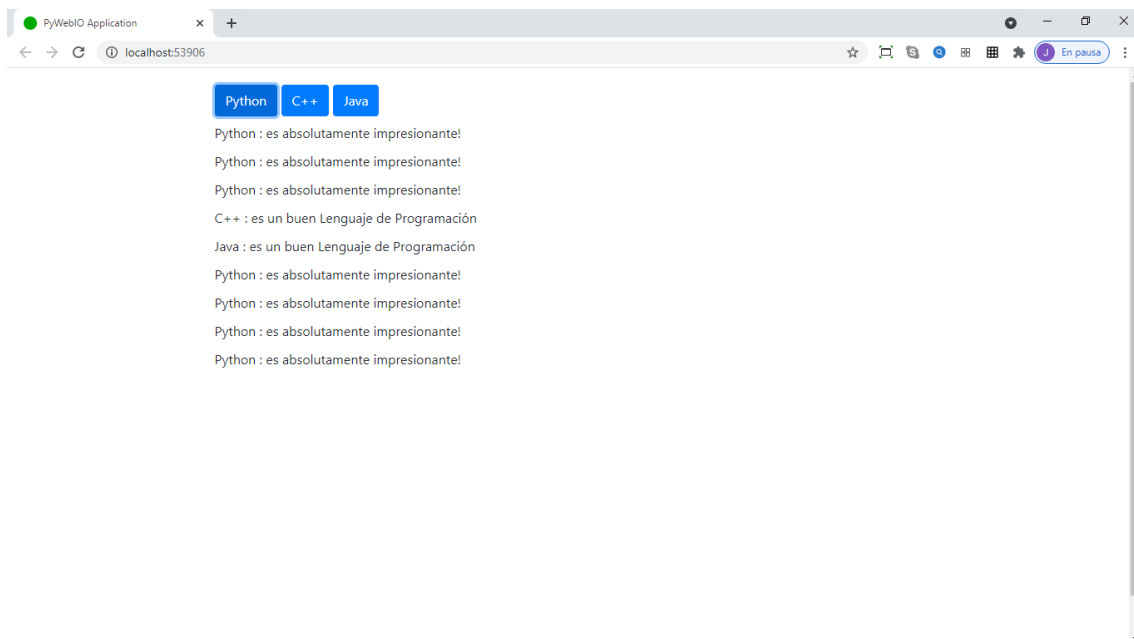


Figura 8.2: Put\_buttons en PyWebIO (parte 2)

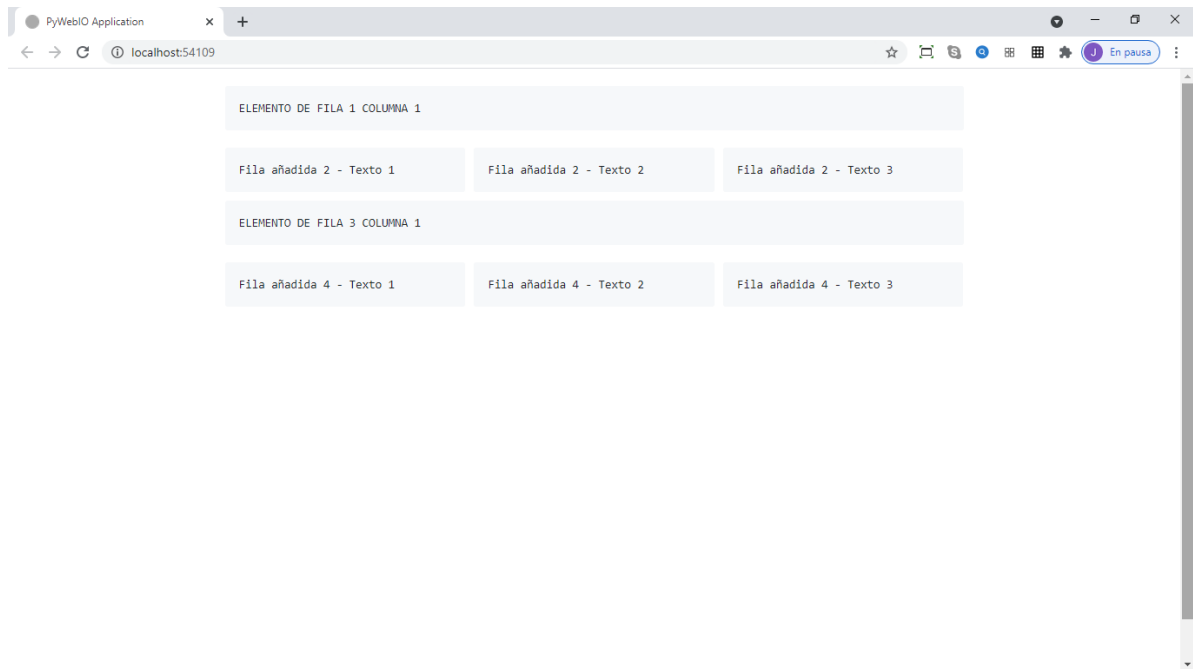
## 9. EJEMPLO 9 CON PYWEBIO : PUT\_ROW / PUT\_COLUMN

```

pywebio_9.py
1  • from pywebio.input import *
2  • from pywebio.output import *
3
4  • put_row([
5  •      put_column([
6  •          put_code('ELEMENTO DE FILA 1 COLUMNA 1'), None,
7  •          put_row([
8  •              put_code('Fila añadida 2 - Texto 1'), None,
9  •              put_code('Fila añadida 2 - Texto 2'), None,
10 •              put_code('Fila añadida 2 - Texto 3')
11 •          ])
12 •      ])
13 •  ])
14 • put_row([
15 •     put_column([
16 •         put_code('ELEMENTO DE FILA 3 COLUMNA 1 '), None,
17 •         put_row([
18 •             put_code('Fila añadida 4 - Texto 1'), None,
19 •             put_code('Fila añadida 4 - Texto 2'), None,
20 •             put_code('Fila añadida 4 - Texto 3')
21 •         ])
22 •     ])
23 • ])

```

Figura 9.1: Put row y Put\_column en PyWebIO (parte 1)



*Figura 9.2: Put row y Put\_column en PyWebIO (parte 2)*

## 10. EJEMPLO 10 CON PYWEBIO : STYLE()

```

pywebio_10.py x
1  • from pywebio.input import *
2  • from pywebio.output import *
3
4
5  # es posible usar style() para hacer una salida customizada
6
7  • style(put_text('Texto en Azul'), 'color: blue')
8
9  • put_table([
10 •     [style(put_text('Texto en negro'), 'color: black'),
11 •       style(put_text('Texto en rojo'), 'color: red')],
12
13 •     [style(put_text('Texto en naranja'), 'color: orange'),
14 •       style(put_text('verde'), 'color: green')],
15 ])

```

Figura 10.1: Style() en PyWebIO (parte 1)

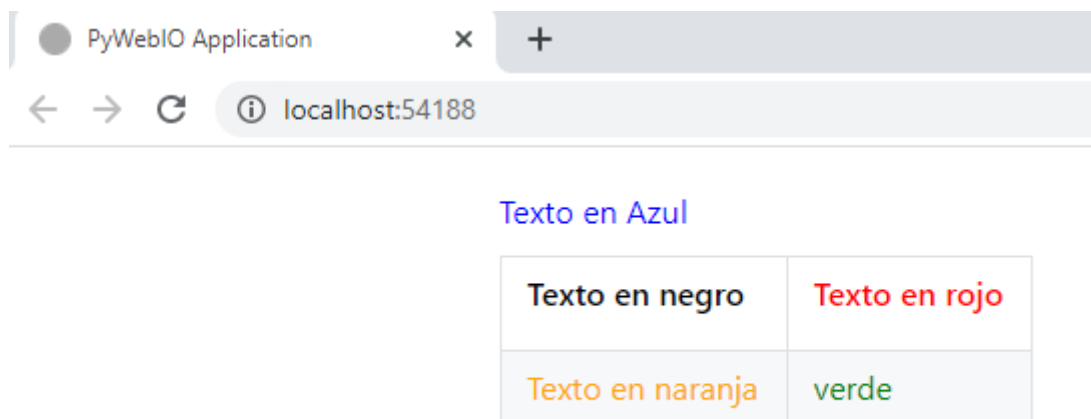


Figura 10.2: Style() en PyWebIO (parte 2)

## 11. OTROS PUNTOS A TRATAR Y CIERRE DE LA ASIGNATURA

La tecnología avanza muy rápidamente y, es posible que en los próximos meses (incluso semanas) salgan nuevos Frameworks innovadores.

Es imprescindible para estar actualizado aprender nuevas cosas (casi constantemente).

A día de hoy (Abril 2021) la tendencia generalizada es conceder importancia a FastAPI, Dash, Streamlit y Django principalmente, así como alguna de las opciones explicadas en Frameworks GUI (Tkinter y PyQt).

Quizá “mañana” existirán otros Frameworks más interesantes o utilizados.

Respecto a otros Frameworks, es conveniente comentar que existen otras cosas que podrían ser utilizadas en otros módulos relacionados con Big Data y Machine Learning. (de hecho sería ideal seguir repasando cosas de esta asignatura en futuras materias, o incluso aprendiendo cosas nuevas).

Finalmente, y con la idea de no abrumar al estudiante con tanto contenido no se ha incluido contenido relacionado con PyCaret (AutoML), Bokeh, e incluso una explicación más detallada de las Series Temporales con Arima, o “Prophet” (Facebook Prophet), por ejemplo.

(Puede que en Big Data y/o Machine Learning sea posible).

Lo que sí esperamos es que al concluir esta materia tengas muy claro en la cabeza si prefieres aprender más sobre un Framework concreto y dedicarte profesionalmente a ello, y que sepas que es posible que, incluso siendo “Data Scientist” o “Software Developer” es muy probable que tengas que trabajar con Aplicaciones.

¡ Mucho ánimo con tu aprendizaje futuro y Muchas Gracias por tu atención !

## 12. PUNTOS CLAVE

- | PyWebIO nos permite hacer aplicaciones de forma rápida, sencilla y de una forma bastante flexible.
- | En el futuro será imprescindible (para ser innovador(a) ) tratar de estar actualizado a nuevas tendencias, tecnologías, Frameworks, etc,. Pero por el momento, los próximos 6-18 meses, probablemente sea suficiente sabiendo bien alguna(s) de las cosas que hemos planteado.

